Available online at www.sciencedirect.com

ScienceDirect

Karbala International Journal of Modern Science 1 (2015) 15–25

http://www.journals.elsevier.com/karbala-international-journal-of-modern-science/

On the designing of two grains levels network intrusion detection system

Safaa O. Al-mamory [a,*], Firas S. Jassim [b]

[a] College of Business Informatics, University of Information Technology and Communications, Iraq

[b] College of Sciences, University of Dyala, Iraq

Received 5 June 2015; revised 27 July 2015; accepted 29 July 2015

Available online 26 September 2015

Abstract

Despite the rapid progress of the information technology, protecting computers and networks remain a major problem for most authors. In this paper, two grains levels intrusion detection system (IDS) is suggested (*fine-grained* and *coarse-grained*). In normal case, where intrusions are not detected, the most suitable IDS level is the *coarse-grained* to increase IDS performance. As soon as any intrusion is detected by *coarse-grained* IDS, the *fine-grained* is activated to detect the possible attack details. Very fast decision tree algorithm is used in both of these detection levels. In order to ensure efficiency of the proposed model, it has been tested on KDD CUP 99 offline dataset and a real traffic dataset. Experimental results demonstrate that the proposed model is highly successful in detecting known and unknown attacks, and can be successfully adapted with packets' flow to increase IDS performance. This article explains how we got a detection rate greater than 93% with an average processing time equals to 3×10^{-6} s per example.

Keywords: Network security; Intrusion detection system; Classification; Very fast decision tree algorithm

1. Introduction

The frequency of computer intrusions has increased rapidly during the last two decades. Intrusion Detection Systems (IDSs) are an essential component of a complete defense-in-depth architecture for network security. They collect and inspect packets, looking for evidence of intrusive behaviors. As soon as an intrusive event is detected, an alarm is raised giving the security analyst an opportunity to react promptly. Unfortunately, most of designed IDSs cannot cope with fast networks.

Although several IDS systems are available, the common objectives of these systems are to reduce the amount of false alarms [1], and to recognize new attacks in order to increase detection ratio. In this paper, the concentration is on detecting known and unknown attacks in fast networks in order to mitigate the influence of the attack by shrinking the time gap between the real attack and its detection.

This paper contribution is to build two grains levels IDS in order to detect abnormal behavior of network traffic and cope with fast networks. It is well known that the intrusion occurrence in networks with respect

* Corresponding author.

E-mail address: salmamory@uoitc.edu.iq (S.O. Al-mamory).

Peer review under responsibility of University of Kerbala.

http://dx.doi.org/10.1016/j.kijoms.2015.07.002

to normal traffic is rare. These motivate us to build the proposed two grains levels IDS. These detection levels are *fine-grained* and *coarse-grained*. In normal case, where intrusions are not detected, the most suitable IDS level is the *coarse-grained* to increase monitoring performance. At the moment of intrusion is detected by *coarse-grained* IDS, the *fine-grained* IDS is activated to detect as most as possible of attack details. Fig. 1 shows the main idea. The *coarse-grained* IDS focuses on five packet features while *fine-grained* IDS works on 20 features. Very Fast Decision Tree (VFDT) [2] algorithm is selected as a fast classifier. The advantages of the proposed system are processing and analyzing of high-speed network traffic, discovering and accurately identifying new attacks to reduce the false alarms to the maximum extent, and detecting the intrusion in real time.

DARPA KDD CUP 99 dataset is used as a benchmark for the proposed IDS, which contains 41 features. As a preprocessing step, we analyzed these features and have selected 20 features having information gain ratio over the average of the dataset. Then, we trained and tested the proposed system. This gave us a detection rate greater than 93% with an average processing time equals to 3×10^{-6} s per example.

This paper is organized as follows: Section 2 reviews related work. Section 3 describes very fast decision tree algorithm. Section 4 states the proposed system. Section 5 presents the experiments and results. Finally, Section 6 concludes this paper.

2. Related work

Nowadays, authors have designed numerous IDSs to detect computer and network intrusions. Several data mining techniques have been used to make networks' intrusions detectable. The first class of approaches uses decision trees (DT) to build attack model. Several variations of decision trees were used such as partial decision tree [3], C4.5 [4], random forest [5], ID3 decision tree [6], and J48 [7]. These decision trees models vary in the splitter measure (i.e. information gain, gain ratio, gini index), pruning technique, branching types, dataset types, etc. The common objective of these decision trees is to iteratively partition the given dataset into subsets where all elements in each final subset belong to the same class. These models have been built from network packets to detect network intrusions with high precision. The main issue with these methods is that they cannot be adaptive with distribution variation in network packets while the proposed system solved this problem by selecting algorithm which works with concept drift.

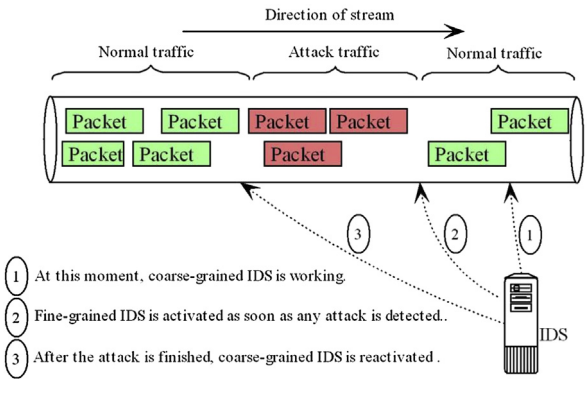

Fig. 1. States the working of the proposed system.

Another class of these approaches has used evolutionary computation [8]. Self-Organizing Map [7] and Multilayer Perceptron (MLP) [7] were trained to recognize normal from abnormal traffic. In addition, genetic programming [9] is achieved very high detection ratio combined with slow model. However, these techniques have performance issues and cannot work in online mode. One of the main goals of this paper is to enhance IDS performance.

Different class of efficient data mining approaches is used to differentiate malicious traffic from normal ones. Bayes network classifier by Staniford et al. [10] is used to calculate the conditional probabilities of several connection features with respect to other connection features. The anomalous connection is determined using these probabilities. SVM is used by Eskin et al. [11], and Honig et al. [12] in addition to their clustering methods for unsupervised learning. The achieved performance was as good as or better than both of their clustering methods. In addition, Fuzzy logic rules by Luo [13] attempted to classify network data. The author verified that the combination of fuzzy logic with association rules and frequency episodes generates more abstract and flexible patterns for anomaly detection. The author approach utilizes fuzzy association rules and fuzzy frequency episodes to extract patterns for temporal statistical measurements at a higher level than the data level.

An additional class of approaches proposed Multi-level IDSs to achieve highest attack detection rate. Multi-level IDS designed by Chen et al. [14] is composed of IDS, firewall, and a report system in order to present a unified report format to the end user. This multi-level IDS supports specific types of these

integrated systems. This system focuses on reporting technique which is different from ours. The most related work to ours is the multi-level IDS (ML-IDS) by Al-Nashif [15] that uses autonomic computing to automate the control and management of ML-IDS. Three levels of granularities are used by ML-IDS which are traffic flow, packet header, and payload. Then it employs a fusion decision algorithm to improve the overall detection rate and minimize the occurrence of false alarms. Genetic algorithm, neural network, least square and other approaches have been used in multiple-level decision fusion, which are different from the used technique in the proposed system. In addition, the proposed system focuses on designing lightweight IDS while ML-IDS goal is to be autonomic.

3. Very fast decision tree algorithm

VFDT is a high-performance data mining system based on Hoeffding trees. Many of classification learning methods have been proposed, of which the decision tree learning method is commonly used. This is because it is fast and the description of classifiers that it derives is easily understood. One of the data stream algorithms that support the decision tree learning method is the VFDT. As data arrives, this data stream grows gradually while the data is classified [16]. VFDT allows the use of either information gain or the Gini index as the attribute evaluation measure. It includes a number of refinements to the algorithm [2]. The pseudo code of VFDT is revealed in Fig. 2.

Algorithm VFDT (S,X,G,δ)
Inputs: S is a sequence of vector of features , **X** is a set of discrete attributes, **G(.)** is a split evaluation function, **δ** is one minus the desired probability of choosing correct feature at any given node.
Output :HT is a decision tree.
Begin
1: Let HT be a tree with a single leaf l_1 (the root).
2: Let $X_1 = X \cup \{X_0\}$.
3: Let $\overline{G_1}(X_0)$ be the $\overline{G}$ obtained by predicting the most frequent class in S.
4: For each class y_k
5: For each value x_{ij} of each attribute $X_i \in X$
6: Let $n_{ijk}(l_1) = 0$.
7: For each example (x, y_k) in S
8: Sort (x, y) into a leaf l using HT.
9: For each x_{ij} in x such that $X_i \in X_l$
10: Increment $n_{ijk}(l)$.
11: Label l with the majority class among the examples seen so far at l.
12: If the examples seen so far at l are not all of the same class, then
13: Compute $\overline{G_l}(X_i)$ for each attribute $X_i \in X_l - \{X_0\}$
14: Using the counts $n_{ijk}(l)$.
15: Let X_a be the attribute with highest G_l.
16: Let X_b be the attribute with second-highest G_l.
17: Compute $\varepsilon = \sqrt{\frac{(R^2 \ln(1/\delta))}{2n}}$
18: If $\overline{G_l}(X_a) - \overline{G_l}(X_b) > \varepsilon$ and $X_a \neq X_0$ then
19: Replace l by an internal node that splits on X_a.
20: For each branch of the split
21: Add a new leaf lm, and let $X_m = X - \{X_a\}$.
22: Let $\overline{G_m}(X_0)$ be the $\overline{G}$ obtained by predicting the most frequent class at l_m.
23: For each class y_k and each value x_{ij} of each attribute $X_i \in X_m - \{X_0\}$
24: Let $n_{ijk}(l_m) = 0$.
25: Return **HT**.
End

Fig. 2. The VFDT algorithm [2].

The VFDT does not accumulate the examples in main memory. The reason for that is it can gradually grow without waiting for the arrival of all the examples. The construction algorithm of the VFDT accumulates only the classes of examples and the contemporaneous occurrence frequency of attribute values in each node to decrease the consumption of memory and processing time, instead of accumulating examples in a decision tree [2].

The VFDT gradually grows as examples are received to create leaf nodes that grow into branches from only the root node. When it creates new nodes, it grows the decision tree, accumulating frequency information in the previous node and measuring whether the new nodes fulfill the statistical criteria [16].

VFDT is different from classical decision trees algorithms. Classical decision tree receives all examples as input and is working in an offline mode. Therefore, it cannot be applied to data streams. On the other hand, a VFDT construction in which new examples arrive in sequence at short intervals in a data stream and huge number of accumulated examples is called an online type decision tree [16].

Very fast decision tree can optionally decide that there is effectively a tie and split on the current best attribute if $\Delta G < \varepsilon < \tau$, where τ is a user-specified threshold. The most significant part of the time cost per example is re-computing G. It is inefficient to recomputed G for every new example, because it is unlikely that the decision to split will be made at that specific point. Thus, VFDT allows the user to specify a minimum number of new examples n_{min} that must be accumulated at a leaf before G is recomputed. This effectively reduces the global time spent on G computations by a factor of n_{min}, and can make learning with VFDT nearly as fast as simply classifying the training examples. Memory usage is also minimized by dropping early on attributes that do not look promising. As soon as the difference between an attribute's G and the best one's becomes greater than ε, the attribute can be dropped from consideration, and the memory used to store the corresponding counts can be freed. VFDT can rescan previously-seen examples. This option can be activated if either the data arrives slowly enough that there is time for it, or if the dataset is finite and small enough that it is feasible to scan it multiple times. This means that VFDT needs never grow a smaller (and potentially less accurate) tree than other algorithms. This is because of using each example only once [2]. A simple comparison between DT and VFDT is based in Table 1.

Table 1
A comparison between DT and VFDT.

Algorithm parameter	DT	VFDT
Time	Slow	Very Fast
RAM utilization	High	Low
Accuracy	Low	Average
Concept-drifting	No	No
Classification error	High	Average
Hoeffding bounded	No	Yes
Number nodes	High	Average
Data stream	No	Yes
Rescan	Never	Low
Tree update	No	Yes
Computational	Average	Low
Data staying	Memory	Online
No. example for learning	Limited	Not limited
No. of parameters	3	4

4. The proposed system

The proposed system is dedicated to detect intrusions on a network by using anomaly intrusion detection approach. This approach is used to detect the known and novel attacks in traffic network. The proposed system operates in two grains levels. The first one works with basic features while the second mode works with statistical features of captured packets. Definition 1 states these features; more description on these features can be found in Ref. [17]. Definition 2 states the two grains levels IDS.

Definition 1. The basic features set *BF* is a five tuples set such that *BF* = <source IP, destination IP, source port, destination port, protocol>. The statistical features set *SF* is a 20 tuples set such that *SF*=<duration, protocol_type, service, flag, src_bytes, dst_bytes, logged_in, count, srv_count, serror_rate, srv_serror_rate, rerror_rate, srv_rerror_rate, same_srv_rate, dst_host_count, dst_host_srv_count, dst_host_same_srv_rate, dst_host_serror_rate, dst_host_srv_serror_rate, dst_host_srv_rerror_rate>. *BF* can be directly extracted from any packet while *SF* represents a connection and requires a set of packets to be created.

Definition 2. *A fine-grained* IDS is the system working on SF set of features. *Coarse-grained* IDS is the system working on *BF* set of features.

The proposed system is composed of two grains levels IDS that allows the system to analyze network traffic on different granularities. The two levels IDS is different from available IDSs in which it adapts with network situation when it is under attack or not. Its detection levels are *Coarse-grained* IDS and *fine-grained* IDS. In normal case, where intrusions are not

detected, the most suitable level is the *Coarse-grained* IDS where five features are monitored to increase IDS performance. At the moment of intrusion is detected by *Coarse-grained* IDS, the *fine-grained* IDS is activated where 20 features are monitored to detect as most as possible of attack details. VFDT algorithm is selected as classifier to achieve this goal because it is capable of processing and analyzing of high-speed network traffic, and detecting the intrusion in real time. The pseudo code of the proposed system is presented in Fig. 3.

The switching between two grains levels of IDS can be noted in Fig. 3. Lines (5–10) represent the *coarse-grained* IDS where five standard features will be inspected. The more packets' features inspected are the more accuracy we will get. For this reason, the *fine-grained* IDS (Lines 13–18) is proposed to inspect more information about packets. The extracted features are organized as connection records to feed them to the model directly in order to determine the nature of connection as soon as possible to give the appropriate response. The *fine-grained* IDS gives more accurate results than *coarse-grained* IDS but at the expense of response time, since it waits enough number of packets to calculate the connection record.

The proposed system consists of four processing stages, which are data collection, pre-processing, classification, and response. Both IDS levels require adequate connection information to train the proposed model. Therefore, the system is doing update the information at any time to obtain a sufficient number of connections which enable building a decision tree for attacks. After connection/packet information is available, a VFDT algorithm is applied in one of the IDS levels in order to do a classification and make the decision (either normal or attack). In case of an attack is detected, a report is generated providing information of the attacks, e.g. IP addresses, time … etc.

The main task of the two grains level IDS is to identify intrusion patterns by considering the features that are extracted from packets. The VFDT detector constructs a decision tree by using constant memory and constant time per sample. The tree is built by recursively replacing leaves with decision nodes. Sufficient statistics of attribute values are stored in each leaf. Heuristic evaluation function is used to determine split attributes converting from leaves to nodes. Nodes contain the split attributes and leaves contain only the class labels. The leaf represents a class that the sample labels. When a sample enters, it traverses the tree from root to leaf, evaluating the relevant attribute at every single node. After the sample reaches a leaf the existing statistics are updated. At this time, the system evaluates each possible condition based on attribute values: if the statistics are sufficient to support the one test over the other, a leaf is converted to a decision node. The decision node contains the number of possible values for the chosen attribute of the installed split test. A decision tree to classify the attacks may become large. However, the error rate is high because it becomes very complicated tree. Therefore, decision tree is pruned to minimize error rate and becomes more simply and easily understandable.

Algorithm Two-levels IDS
Input: Stream of packets
Output: Alerts if any attacks are detected

```
Begin
 1:  Mode=H_Level
 2:  While (P=packet capturing()){
 3:    Preprocess (P)
 4:    if (Mode=H_Level)
 5:          // Coarse-grained IDS
 6:          E= BF(P)
 7:          Classify E using VFDT model
 8:          if (any attack is detected){
 9:                Mode=C_Level
10:                Generate Alerts}
11:  else{
12:          // fine-grained IDS
13:          E= SF(P) where E is built from a set of P
14:          Classify E using VFDT model
15:          if (any attack is detected)
16:                Generate Alerts
17:          if (no attack is detected within a specific period)
18:                Mode=H_Level
19:  }
End
```

Fig. 3. The pseudo code for the proposed system.

5. The experimental results

In this section, we describe the experiments conducted to evaluate our system. The proposed system was tested on a P4 Core (TM) i3CPU processor 2.13GHzwith 2.00 GBRAM running Linux Fedora. Two different datasets were used in our experiments: a real dataset, and KDD CUP 99 dataset [17]. The real dataset was collected from a network consisted of three hosts for a week and a rate of 10 h a day. The results with real dataset are not included in this paper for space purposes. In the following subsections, we consider KDD CUP 99 dataset in a level of details, preprocessing stage, the performance metrics, and the main results we got.

5.1. KDD CUP 99 dataset description

Since 1999, KDD CUP 99 [17] has been the most widely used dataset for the evaluation of anomaly detection methods. Furthermore, it contains labeled connections thus facilitates the process of training and testing the model, which encouraged us to use it. It contains about 5 million connection records, each with about 100 bytes. The two weeks of testing data have around two million connection records. KDD CUP 99 training dataset consists of 4,898,430 single connection vectors each of which contains 41 features. These connections are labeled as either normal or an attack, with exactly one specific attack type [18]. KDD CUP 99 is actually composed of three datasets: *Whole KDD* which contains about four million registers, *10% KDD* dataset, and *Corrected KDD* for testing purposes. More description about the dataset is stated in Table 2.

The simulated attacks fall in one of the four categories [19]. These categories are Denial of Service Attack (DoS), Probing Attack, User to Root Attack (U2R), and Remote to Local Attack (R2L). It is important to note that the test data is not from the same probability distribution as the training data, and it includes specific attack types which are not appeared in the training data which make the task more realistic. The datasets contain a total number of (22) training attack types, with an additional (16) types in the test data.

The dataset has 41 attributes for each connection record plus one class label. Features are grouped into four categories [19]. Firstly, basic features which can be derived from packet headers without inspecting the payload. Secondly, content features in which the domain knowledge is used to assess the payload of the original TCP packets. Thirdly, time-based traffic features are designed to capture properties that mature over a 2 s temporal window (Statistical). Finally, host-based traffic features utilize a historical window estimated over the number of connections.

5.2. Preprocessing

Different preprocessing techniques have been applied on KDD CUP 99 dataset one of which is features selection. Irrelevant and redundant features decrease not only the detection speed but also detection accuracy possibly. The *information gain* (IG) measure had been used to rank these feature. Fig. 4 shows the classification of the (41) features of the KDD CUP 99 dataset sorted in a descending order by information gain ratio. Most of the features have information gain under the average of the dataset, (IG average = 0.22). In fact, only 20 features are above IG average. This shows that the original database has data concentration in a small group of values. Features that result in a convergence of connection categories within a small group of values are less important for describing a node behavior. This indicates that the original dataset may contain irrelevant data for the IDS and so needs to be optimized. Another reason for selecting 20 features can be concluded from Fig. 5 which shows the relation among system accuracy and execution time required for training the model, as well as number of selected features in training phase.

Another pre-processing has been done by converting each feature from text or symbolic into numerical form. In this conversion, an integer code is assigned for each symbol. For instance, in the case of protocol type feature, 0 is assigned for TCP, 1 for UDP, and 2 for the ICMP symbol. Attack names were first mapped to one of the twenty-three classes, 0 for Normal, until 22for the warezmaster attack.

Table 2
Number of samples in KDD CUP 99 datasets.

KDD dataset	Total	DoS	Probe	R2L	U2R	Normal
Whole KDD	4,898,430	3,883,370	41,102	1126	52	972,780
Corrected KDD	311,029	229,853	4166	16,347	70	60,593
10% KDD	494,020	391,458	4107	1126	52	97,277

5.3. Performance metrics

For ranking the different results, there are standard metrics that have been used in order to evaluate network intrusion detections. Detection Rate (DR) and False Alarm Rate (FAR) are the most two famous metrics that have already been used. Detection rate metric is computed as the ratio between the number of correctly detected attacks and the total number of attacks as in Equation (1). FAR metric is computed as the ratio between the number of normal connections that are incorrectly misclassified as attacks and the total number of normal connections; see Equation (2) [20].

$$DR = \frac{Number\ of\ Detected\ Attacks\ Correctly}{Total\ Number\ of\ Attacks} \quad (1)$$

$$FAR = \frac{False\ Positives}{Total\ Number\ of\ Normal\ Connections} \quad (2)$$

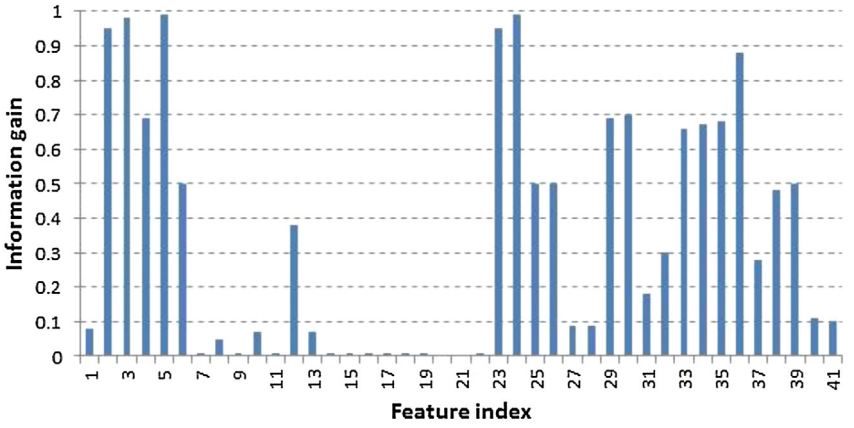

Fig. 4. The maximum information gain for each feature.

5.4. Results

The application of VFDT algorithm is not only for distinguishing attacks from normal behaviors, but also identifies different types of intrusions. During training phase, the VFDT constructs a model from selected feature of offline KDD CUP 99. The VFDT was trained on normal and attack traffic in order to classify network traffic according to the range of values in each features connection at the data link, network and transport layers. The file kdd cup.data.gz that contains the full dataset is used for training model.

The corrected.gz file from KDD CUP 99 dataset is used for testing the model. It contains data with corrected labels to evaluate the VFDT model. This data contains more types of attacks than the KDD CUP 99 training set (i.e. 16 new attacks out of 38 possible attacks). Thus, the VFDT can be tested with new types of attacks, which means that the VFDT can be tested as the anomaly detection method. Table 3 shows numbers and percentage training and testing vectors used in the proposed model. From this table, we can note that some attacks are rare with respect to others. This adds a challenge to the proposed system.

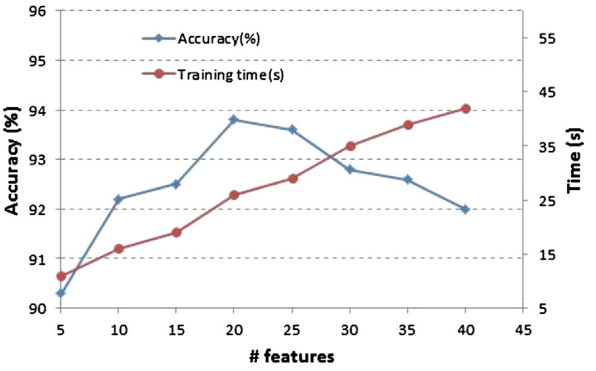

Fig. 5. The effect of the number of selected features on accuracy and training time.

Table 4 presents the confusion matrix (CM) related to the standard metric for the proposed system. The number of successful predictions as normal examples was (41,951) of the total (42,187), while the model failed in detecting (1631) examples. Furthermore, the number of successful predication for attack examples was (23,870) from the total (25,501), while false alarm was (236) examples.

As a classifier for five classes, VFDT algorithm produced the results shown in Table 5. Then, the cost per training example for this CM computation was (0.1179). In this Table, the detection rate for U2R and R2L was zero. This is because the used number of connections to train the model on these attacks was very small compared with the rest of the other types of attacks. This case, detection of rare classes, is a challenge for most existing classifiers. In addition, VFDT has good detection for normal, DoS, and Probe attacks, because number of connections that have been used for the training on these classes was appropriate to identify classified.

Accuracy is the main comparison measure for the classifiers. VFDT has competitive accuracy performance when compared with other classification algorithms, where the classification accuracy was (93.825%). The ROC curve of our system is shown in Fig. 6, our goal is thus to detect as many attacks as possible while minimizing the generation of false alarms. Fig. 6 shows that our system was able to detect most of the attacks for the KDD CUP 99 data at FAR about (0.608%).

Table 3
Number of train and test vectors for all classes with the percentage.

Class name	Whole KDD CUP 99 dataset		Correct test data	
	Number	Percentage	Number	Percentage
Normal	812,814	75.6	42,187	62.3
Back	968	0.09	386	0.58
Land	19	0.002	9	0.013
Neptune	942,149	22.5	20,227	29.89
Pod	206	0.02	45	0.066
Smurf	3007	0.3	939	1.38
Teardrop	918	0.08	12	0.017
Satan	5012	0.5	1599	2.36
Port_sweep	3564	0.3	108	0.15
Nmap	1554	0.14	80	0.159
IPsweep	3723	0.35	155	0.228
Load_module	9	0.0008	2	0.003
Buffer_overflow	30	0.0027	22	0.032
Spy	2	0.0002	2	0.003
Perl	3	0.0002	2	0.003
Rootkit	10	0.0008	13	0.019
Ftp_write	8	0.0008	3	0.004
Imap	12	0.001	11	0.016
Phf	4	0.0003	2	0.003
Guess_passwd	53	0.005	1257	1.85
multihop	7	0.0006	18	0.026
Warezclient	893	0.08	91	0.134
Warezmaster	20	0.002	518	0.765
Total	1,774,985	100	67,688	100

Table 6 presents the CM for all classes (normal and different types of attack) when using VFDT algorithm as a classifier. From this table, the normal class has the best detection rate compared with other classes, decreasing the FAR of the model. Furthermore, some classes (attacks) have low detection rate such as (spy, perl, rootkit, ftp_write, phf, multihop, warezclient, and warezmaster), due to the following two reasons. Firstly, the small percentage of these classes in the training is noticed. Secondly, the similarity is high between the connections of these classes with normal class.

Table 7 presents the CM for new attacks; the total number of new connections was (3619). Most of these connections were DoS and Probe connections. The majority of testing data was Probe connections making the VFDT classifier a little bit biased toward the major class. Because of the distribution of the testing data is different from the training data, this makes it difficult

Table 4
CM for VFDT algorithm.

Actual	Predicted	
	Normal	Intrusion (Attack)
Normal	TN (41,951)	FP (236)
Intrusion (Attack)	FN (1631)	TP (23,870)

Table 5
CM for VFDT algorithm to five classes.

	Normal	DoS	Probe	U2R	R2L	Total
Normal	41,951	139	97	0	0	42,187
DoS	73	21,492	53	0	0	21,618
Probe	383	222	1337	0	0	1942
U2R	41	0	0	0	0	41
R2L	1134	762	4	0	0	1900
Total	43,582	22,615	1491	0	0	67,688

to classify. The DoS attack technique in the testing data is different from the one used in the testing data making a low detection rate. Detection rate for DoS Attacks (back, smurf, and neptune attacks) has more than (90%), however other DoS attacks has lower DR, as we can see in Table 6.

Most of the KDD CUP 99 dataset contain the packets that used TCP protocol and the least of those that use ICMP and UDP protocol, this explains the reason when high rate is seen in detecting some attacks such as (smurf, Neptune, back, ..etc.) due to they use TCP protocol, while few percentage in detecting some of the attacks such as (pod, land, teardrop ..etc.) due to it used packets UDP and ICMP protocols.

Table 7 presents the new kinds of DoS attacks such as (mailbomb, process table, UDP storm, and Apache2 attacks) which are rarely detected because the patterns of the encoded data are very different from the patterns of the old DoS attacks.

Detection rate for Probe Attacks, both old (Satan, Port_sweep, Nmap) and new kinds of attacks (Saint, Mscan) are detected with relatively high detection rate. The data size of the probe attacks are bigger than the attacks included in the other attack classes, so that the patterns of probe attack class are more various than the others. This means that numerous patterns of data can be provided as the learning dataset. This is the reason why probe attacks can be detected with the high detection rate.

Execution time and memory consumption are good comparison measures between classifiers. During the

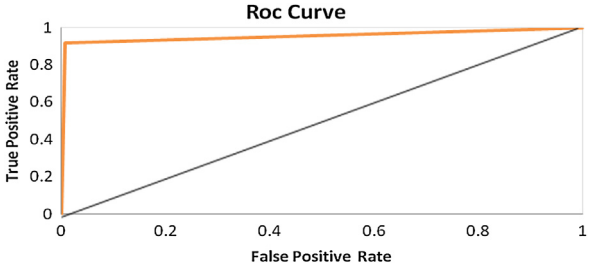

Fig. 6. ROC curve for VFDT Algorithm.

Table 6
CM for VFDT algorithm to all classes.

	Normal	Back	Land	Neptune	Pod	Smurf	Teardrop	Satan	Port_sweep	Nmap	Ipsweep	Load_module	Buffer_overflow	Ftp_write	Guess_passwd	Multihop	Warezclient	Warezmaster	Total
Normal	41,951	3	0	4	0	132	0	0	48	0	49	0	0	0	0	0	0	0	42,187
Back	3	330	0	0	0	0	0	0	53	0	0	0	0	0	0	0	0	0	386
Land	0	0	0	9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	9
Neptune	50	0	0	20,176	0	1	0	0	0	0	0	0	0	0	0	0	0	0	20,227
Pod	5	0	0	0	0	40	0	0	0	0	0	0	0	0	0	0	0	0	45
Smurf	3	0	0	0	0	936	0	0	0	0	0	0	0	0	0	0	0	0	939
Teardrop	12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	12
Satan	383	0	0	204	0	0	0	1012	0	0	0	0	0	0	0	0	0	0	1599
Port_sweep	0	0	0	14	0	0	0	0	94	0	0	0	0	0	0	0	0	0	108
Nmap	0	0	0	0	0	0	0	0	0	80	0	0	0	0	0	0	0	0	80
Ipsweep	0	0	0	0	0	4	0	0	0	0	151	0	0	0	0	0	0	0	155
Load_module	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2
Buffer_overflow	22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	22
Spy	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2
Perl	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2
Rootkit	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13
Ftp_write	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3
Imap	6	0	0	1	0	0	0	0	0	4	0	0	0	0	0	0	0	0	11
Phf	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2
Guess_passwd	496	0	0	761	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1257
multihop	18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	18
Warezclient	91	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	91
Warezmaster	518	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	518
Total	43,582	333	0	21,169	0	1113	0	1012	195	84	200	0	0	0	0	0	0	0	67,688

Table 7
CM for VFDT algorithm to predict new attacks.

	Normal	Back	Land	Neptune	Pod	Smurf	Teardrop	Satan	Port_sweep	Nmap	Ipsweep	Load_module	Buffer_overflow	Rootkit	Ftp_write	Guess_passwd	Multihop	Warezclient	Warezmaster	Total
Mail_bomb (DoS)	308	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	308
Udp storm (DoS)	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2
Process_table (DoS)	741	0	0	0	0	0	0	0	3	0	0	0	0	0	0	0	0	0	0	744
Saint (Probe)	37	0	0	105	0	1	0	492	0	0	101	0	0	0	0	0	0	0	0	736
Mscan (Probe)	359	0	0	694	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1053
Xterm (U2R)	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13
Http_tunnel (U2R)	23	0	0	0	0	0	0	133	2	0	0	0	0	0	0	0	0	0	0	158
Worm (U2R)	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2
Sqlattack (U2R)	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2
Ps (U2R)	16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	16
Snmpguess (U2R)	359	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	359
Snmpgetattack (R2L)	179	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	179
Named (R2L)	11	0	0	6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	17
Xlock (R2L)	9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	9
Xsnoop (R2L)	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4
Sendmail (R2L)	17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	17
Total	2082	0	0	805	0	1	0	625	5	0	101	0	0	0	0	0	0	0	0	3619

Table 8
Results comparison of the proposed system with other systems.

Algorithm	Size of training datasets	Size of testing datasets	DR (%)	Train time (TT) (sec.)	Average training time per example (sec.)
Support vector machine [21]	1,132,365	73,247	57.6	62424	18.14
Genetic programming [9]	24,780	311,028	98	6480	0.2615
Artificial neural networks [8]	4947	3117	92.27	780	0.1576
Multilayer perceptron [7]	49,596	15,437	92.03	350.15	0.007
Self-organizing map [7]	49,596	15,437	91.65	192.16	0.0038
Multivariate adaptive regression splines [22]	11,982	11,982	96.46	30.66	0.0025
Fuzzy logic [23]	54,226	56,226	91.25	87.9	0.0016
Naïve bayes [10]	65,525	65,525	95	1.89	0.0013
C4.5 [4]	49,596	15,437	92.06	15.85	0.0003
J48 [7]	49,596	15,437	92.06	15.85	0.00003
LBK [7]	49,596	15,437	92.22	10.63	0.00002
Incremental tree induction [24]	169,000	311,029	92.38	18	0.00002
Partial decision tree [3]	444,458	49,384	46.67	48.8	0.00002
Random forest [5]	65,525	65,525	90.08	129	0.00002
K-means [25]	55,000	25,000	86	13	0.00002
Bayes Net [7]	49,596	15,437	90.62	6.28	0.00001
Apriori [3]	444,458	49,384	87.5	18.94	0.000005
The proposed system	1,074,985	67,688	**93.83**	39.88	**0.000003**

training phase of VFDT algorithm on 100,000 connections, the execution time was (4.09 s), whereas the execution time on 1,000,000 connections was (38.97 s) additional to (0.71 s) prune time, outperforming all classifiers. According to memory consumption, the VFDT takes one third of memory consumed by traditional C4.5 algorithm [4] and half of BayseNet algorithm [7]. The memory allocation for all instances (1,000,000 connections) was (6.72 M), and number of nodes was (112). These properties of VFDT qualifies it to easily work on fast networks.

A comparison of the proposed system with other published systems was conducted as can be seen in Table 8. Three comparison factors are used here which are dataset size, detection rate, and average training time per example. In this table, we have included the size of the training data and testing data to show how much the results are statistically reliable. The systems in Table 8 have descending order with respect to average training time per example. The fastest system in training is our proposed system as can be seen in the table. While other systems got better detection rate like Multivariate Adaptive Regression Splines, Genetic Programming, Naïve Bayes; however, these systems are very slow and not suitable for stream data.

6. Conclusions

Two levels IDS is proposed allowing the system to analyze network traffic on different granularities. It is different from the available IDSs in which it adapts with network situation when it is under attack or not. Its detection levels are coarse-grained IDS and fine-grained IDS. These two detection levels are tested with DARPA 1999 dataset achieving detection rate higher than (93%), outperforming most classifiers. VFDT has proved its efficiency in both generalization tree and new attacks detection. Model building and testing using a VFDT algorithm did not exceed 40 s compared with other systems which exceeded hours in building their models. The detection rate of the VFDT classifier depends on sufficient training data and the right features' set. The most significant features for the two classes Normal and R2L heavily overlap which limit the detection rate of R2L attacks.

References

[1] R. Perdisci, G. Giacinto, F. Roli, Alarm clustering for intrusion detection systems in computer networks, J. Eng. Appl. Artif. Intell. 19 (2006) 429–438.
[2] D. Pedro, H. Geoff, Mining high speed data streams, in: ACM SIGKDD Conference on Knowledge Discovery and Data Mining, 2000, pp. 71–80.
[3] Mohammed M. Mazid, M. Shawkat Ali, Kevin S. Tickle, A comparison between rule based and association rule mining algorithms, in: 3rd IEEE International Conference on Network and System Security, 2009, pp. 452–455.
[4] G. Radhika, S. Anjali, C.J. Ramesh, Parallel misuse and anomaly detection model, Int. J. Netw. Secur. 14 (4) (2012) 211–225.
[5] P. Mrutyunjaya, R.P. Manas, Evaluating machine learning algorithms for detecting network intrusions, Int. J. Recent Trends Eng. 1 (1) (2009).
[6] Dewan M. Farid, H. Nouria, B. Emna, Z.R. Mohammad, M.R. Chowdhury, Attacks classification in adaptive intrusion

detection using decision tree, World Acad. Sci. Eng. Technol. (2010) 27–44.
[7] A.N. Huy, D. Choi, Application of data mining to network intrusion detection: classifier selection model, in: Asia-Pacific Network Operation and Management Symposium, Springer-Verlag, Berlin, Heidelberg, 2008, pp. 399–406.
[8] M. Adnan, B. Abdulazeez, S.I. Adel, Intrusion detection and attack classifier based on three techniques, A Comp. Study. Eng. Technol. J. 29 (2) (2011) 233–254.
[9] M.F. Kamel, B. Aoued, Securing network traffic using genetically evolved transformations, Malays. J. Comput. Sci. 19 (2006) 3–23.
[10] S. Staniford, J.A. Hoagland, J.M. McAlerney, Practical automated detection of stealthy portscans, J. Comput. Secur. 10 (1–2) (2002) 105–136.
[11] E. Eskin, A. Arnold, M. Preraua, L. Portnoy, S.J. Stolfo, A geometric framework for unsupervised anomaly detection: detecting intrusions in unlabeled data, in: D. Barbar, S. Jajodia (Eds.), Data Mining for Security Applications, Kluwer Academic Publishers, Boston, 2002.
[12] A. Honig, A. Howard, E. Eskin, S.J. Stolfo, Adaptive model generation: an architecture for the deployment of data mining based intrusion detection systems, in: D. Barbar, S. Jajodia (Eds.), Data Mining for Security Applications, Kluwer Academic Publishers, Boston, 2002.
[13] J. Luo, Integrating Fuzzy Logic with Data Mining Methods for Intrusion Detection, Mississippi State University, 1999 (Master thesis).
[14] T. Chen, P. Chen, T. Wang, Y. Chiu, S. Lai, Integrated multi-level intrusion detection and report system, in: Proceedings of the Fifth International Conference on Electronic Business, Hong Kong, 2005, pp. 463–469.
[15] Y. Al-Nashif, Multi-level Anomaly Based Autonomic Intrusion Detection System, University of Arizona, 2008 (PhD dissertation).
[16] M. Tatsuya, N. Ayahiko, Detection of fraud use of credit card by extended VFDT, IEEE Trans. Knowl. Data Eng. 21 (11) (2011) 1505–1514.
[17] KDD Cup 99 Intrusion Detection Dataset, 1999. http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html.
[18] T. Mahbod, E. Bagheri, Wei Lu, A.A. Ghorbani, A detailed analysis of the KDD CUP 99 dataset, in: 2nd IEEE Symposium on Computational Intelligence for Security and Defense Application, 3(5), 2009, pp. 322–332.
[19] A. Nelcileno, R. Oliveira, A. Akira Shinoda, B. Bhargava, Identifying important characteristics in the KDD 99 intrusion detection dataset by feature selection using a hybrid approach, in: International Conference on Telecommunications, 2010.
[20] B. Marjan, E. Salahi, M. Khaleghi, An improved intrusion detection technique based on two strategies using decision tree and neural network, JCIT J. 4 (4) (2009) 96–101.
[21] K. Latifur, A. Mamoun, T. Bhavani, A new intrusion detection system using support vector machines and hierarchical clustering, VLDB J. 16 (2007) 507–521.
[22] M. Srinivas, H.S. Andrew, A. Ajith, R. Vitorino, Intrusion detection systems using adaptive regression splines, CoRR J. (2004).
[23] R. Shanmugavadiru, N. Nagarajan, Network intrusion detection system using fuzzy logic, Indian J. Comput. Sci. Eng. 2 (1) (2011) 101–111.
[24] Y. Wei-Yi, An Incremental–learning Method for Supervised Anomaly Detection by Cascading Service Classifier and ITI Decision Tree Methods, National Taiwan University of Science and Technology, Taiwan, 2009 (Master Thesis).
[25] Satinder P. Singh, Data Clustering Using K-Means Algorithm for Network Intrusion Detection, Department of Computer Science and Engineering, Lovely Professional University, 2010 (Master Thesis).